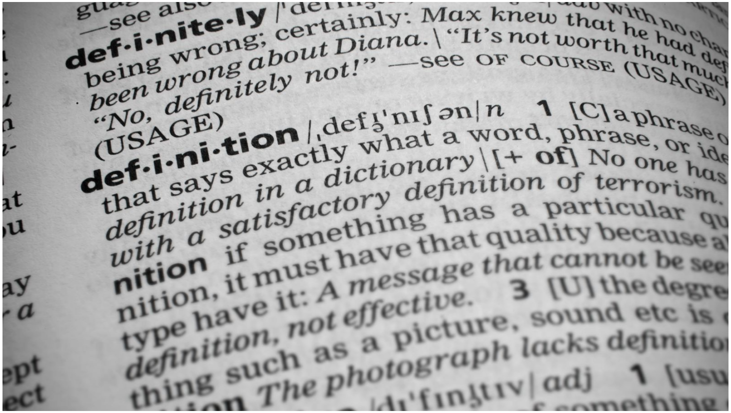
Natural Language Processing

Lecture 4 : Words and Meaning

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Words and meaning



—see also

def·i·nite·ly /ˈde

being wrong; certainly: *Max knew that he had def*

been wrong about Diana. | *"It's not worth that muc*

"No, definitely not!" —see OF COURSE (USAGE)

(USAGE)

def·i·ni·tion /ˌdefɪˈnɪʃən/ *n* **1** [C] a phrase o

that says exactly what a word, phrase, or ide

definition in a dictionary | [+ **of**] *No one has*

with a satisfactory definition of terrorism.

nition if something has a particular qu

nition, it must have that quality because a

type have it: *A message that cannot be see*

definition, not effective. **3** [U] the degre

thing such as a picture, sound etc is c

The photograph lacks definitio

/dɪˈfɪnətɪv/ *adj* **1** [usu

Lexical semantics

The linguistic study of word meaning is called **lexical semantics**.

The **internal** semantic structure of a word refers to the similarity with other words.

Question: how meaningful is it to interchange this word with other words?

The **external** semantic structure of a word refers to the allowability to combine with other words.

Question: how meaningful is it to combine this word with other words?

Example

Contrast the following sentences, where ‘?’ marks weak ungrammaticality and ‘*’ marks strong ungrammaticality

- Alice eats an apple
- ?Alice eats a thunderstorm
  internal violation for word thunderstorm
- *Alice eats an apple to John
  external violation for word eat

Lexical semantics

Lexical ambiguity arises because a word can have different meanings, called **word senses**.

Example :
bank^1: ‘financial institution’
bank^2: ‘sloping mound’

Word sense disambiguation (WSD) is the task of determining which sense of a word is being used in a particular context.

Lexical semantics

Lexical semantic relationship between words are important components of word meaning.

Two words are **synonyms** if they have a common word sense.
Example : **car** and **automobile**

Two words are **similar** if they have similar meanings.
Example : **car** and **bicycle**.

SimLex-999 is a dataset of word similarities for English.

Two words are **related** if they refer to related concepts.
Example : **car** and **gasoline**.

Lexical semantics

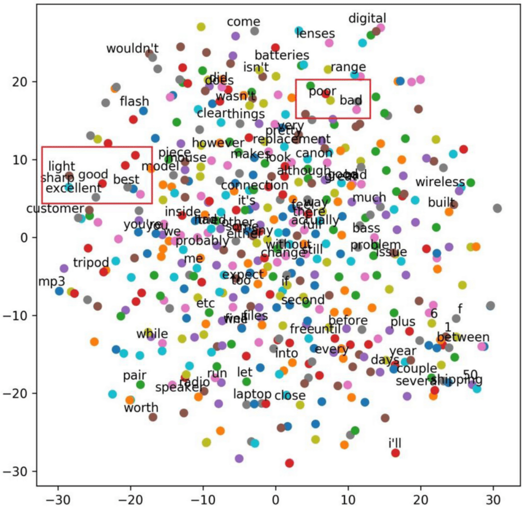
Two words are **antonyms** if they define a binary opposition.
Example : 'hot' and 'cold'.

One word is an **hyponym** of another if the first has a more specific sense. Notions of **hypernym** or **hyperonym** are defined symmetrically.
Example : 'car' and 'vehicle' (subordinate).

Also called subordinate and superordinate relations.

Words can have **affective** meanings, implying positive or negative connotations/evaluation
Example : 'happy' and 'sad'; 'great' and 'terrible'.

Distributional semantics



Distributional semantics

It is very difficult to define the notion of word sense in a way that can be understood by computers.

And also in a way that can be understood by humans!

We take a radically different approach, already foreseen in the following works:

> *The meaning of a word is its use in the language*
>
> *Ludwig Wittgenstein*
> *Philosophical Investigations, 1953*

> *You shall know a word by the company it keeps*
>
> *John Rupert Firth*
> *Selected papers, 1957*

Example

Suppose you don't know the meaning of the word 'tezgüino', but you have seen it in the following contexts

a) A bottle of _________ is on the table.
b) Everybody likes _________.
c) Don't have _________ before you drive.
d) We make _________ out of corn.

What other words fit into these contexts?

	a)	b)	c)	d)
tezgüino	1	1	1	1
motor oil	1	0	0	1
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

Based on these vectors, we conclude that 'wine' is very similar to 'tezgüino'.

Distributional semantics

Distributional semantics develops methods for quantifying semantic similarities between words based on their distributional properties, meaning their **neighboring words**.

The basic idea lays in the so-called **distributional hypothesis**: linguistic items with similar distributions have similar meanings.

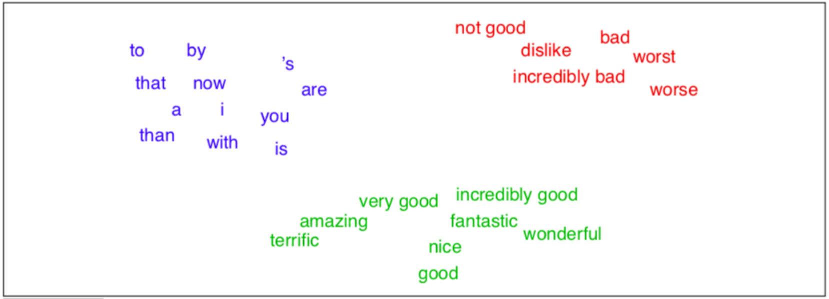
Thus the meaning of a word is defined by its distribution in language use.

The basic approach is to collect distributional information in **high-dimensional vectors**, and to define distributional/semantic similarity in terms of vector similarity.

Distributional semantics

Similar words are mapped into “close enough” vectors.

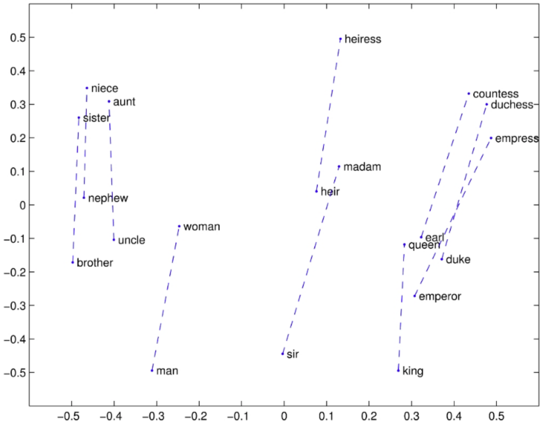
Example : Vectors from sentiment analysis application, projected into 2-dimentional space.



Distributional semantics

Lexical semantic relationships, represented as word vector differences, are generally preserved.

Example : Vectors projected into 2-dimentional space.



Distributional semantics

The obtained vectors are called **word embeddings**.

Each discrete word is embedded in a continuous vector space.

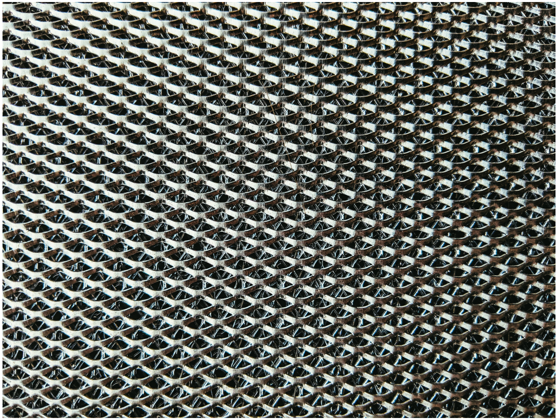
The approach is called **vector semantics** and is the standard way to represent word meaning in NLP. All the learning algorithms we report are **unsupervised**.

Two families of word embeddings:

- **Sparse vectors**: Vector components are computed through some function of the counts of nearby words.
- **Dense vectors**: Vector components are computed through some optimisation or approximation process.

Alternative classifications: frequency-based vs. prediction-based embeddings.

Count-based embeddings


Dimitry Zub on Unsplash

Count-based embeddings

Let $\{w_1, \ldots, w_W\}$ be a set of **term** (target) words and let $\{c_1, \ldots, c_C\}$ be a set of **context** words.

For each term word w_i in the corpus, we consider context words appearing inside a **window** of size L, at the left or at the right of w_i.

A **term-context matrix** F is a matrix of size $W \times C$, where each element $f_{i,j}$ is the number of times context word c_j appears in the context of term word w_i in the corpus.

Example

Term words: ‘cherry’, ‘digital’, ‘information’
Context words: ‘pie’, ‘data’, ‘computer’
Term-context matrix F:

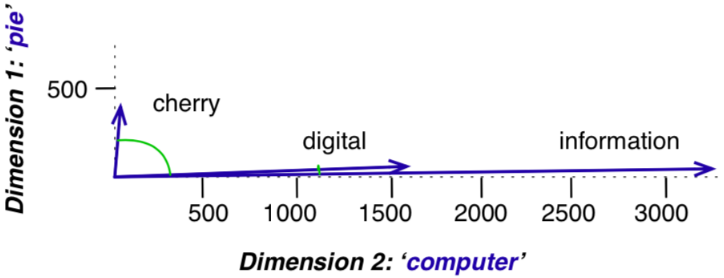
	pie	**data**	**computer**
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

Cosine similarity between vectors associated with term words:

$$\begin{aligned}
\cos(\text{cherry}, \text{information}) &= \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .018 \\
\cos(\text{digital}, \text{information}) &= \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996
\end{aligned}$$

Example

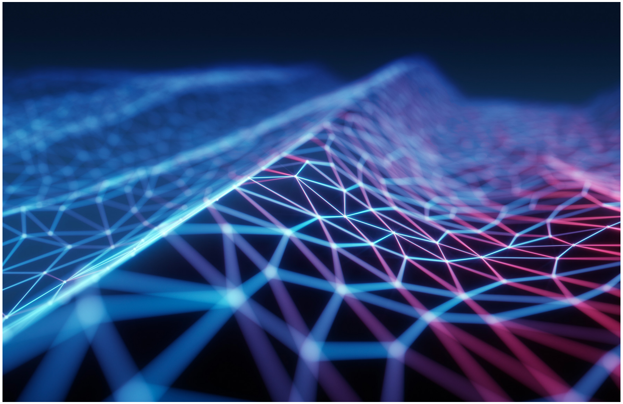
We restrict to dimensions 'pie' and 'computer' for convenience.



The model states that 'information' is way closer to 'digital' than it is to 'cherry'.

Observe that the difference in frequency between 'information' and 'digital' results in an **undesired** difference along the 'computer' dimension.

Neural word embeddings


©STAR7

Neural word embeddings

Word embeddings can also be computed using **neural networks**.

Generally speaking, neural word embeddings have the following properties:

- small number of dimensions, in the range [50..1000]
- dense vectors
- components can be negative
- the dimensions don't have a clear interpretation

Neural word embeddings **work better** than previous embeddings in every NLP task.

Neural word embeddings

The neural word embeddings we introduce in this lecture are **static**, meaning that the presented methods learn one fixed embedding for each word in the vocabulary.

Later we will introduce dynamic (or contextual) embeddings such as BERT embeddings, which depend on specific sentences or documents.

Word2vec

Word2vec is a software package including two different algorithms for learning word embeddings:

- skip-gram with negative sampling (SGNS)
- continuous bag-of-words (CBOW)

We look into skip-gram in detail.

Idea : Instead of counting how often a context word appears near a target word, we train a classifier on the following binary **prediction task**:

> *Is a given context word likely to appear near a given target word?*

We don't really care about this prediction task: instead, we use the learned representation as the word embeddings.

Skip-gram

More specifically, the basic steps of the **skip-gram** algorithm are as follows.

For each target word $w_t \in V$:

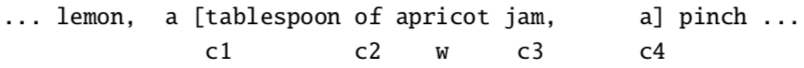
- treat w_t and any neighboring context word w_c as **positive examples**
- randomly sample other words $w_n \in V$, called **noise words**, to produce **negative examples** for w_t

Use **logistic regression** to train a classifier to distinguish positive and negative examples.

Use the learned parameters as the embeddings.

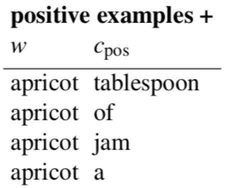
Example

Consider a specific word position in the training text and a context window of size ± 2:



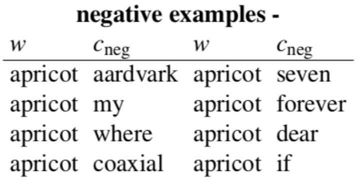
```
... lemon,  a [tablespoon of apricot jam,     a] pinch ...
               c1          c2    w      c3      c4
```

Assume a ratio of 1:2 between positive and negative examples. We get the following examples:



positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a



negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Logistic regression

For any pair of words $w_t, u \in V$, we need to define the probability that u is/is not a context word for target word w_t

$$P(+ \mid w_t, u)$$
$$P(- \mid w_t, u) = 1 - P(+ \mid w_t, u)$$

For each $w \in V$, we construct two complementary **embeddings**

- a target embedding $\mathbf{e}_t(w) \in \mathbb{R}^d$
- a context embedding $\mathbf{e}_c(w) \in \mathbb{R}^d$

Integer d is the size of the embedding. Operator $\mathbf{e}_t(w)$ is always assigned to target words, operator $\mathbf{e}_c(w)$ is always assigned to context or noise words.

Logistic regression

We can now define

$$P(+ \mid w, u) = \sigma(\mathbf{e}_t(w) \cdot \mathbf{e}_c(u)) = \frac{1}{1 + \exp(-\mathbf{e}_t(w) \cdot \mathbf{e}_c(u))}$$

We then have (for fixed vector module)

- vectors $\mathbf{e}_t(w), \mathbf{e}_c(u)$ with small angle: positive cosine similarity, $P(+ \mid w, u)$ close to one
- vectors $\mathbf{e}_t(w), \mathbf{e}_c(u)$ with large angle: negative cosine similarity, $P(+ \mid w, u)$ close to zero

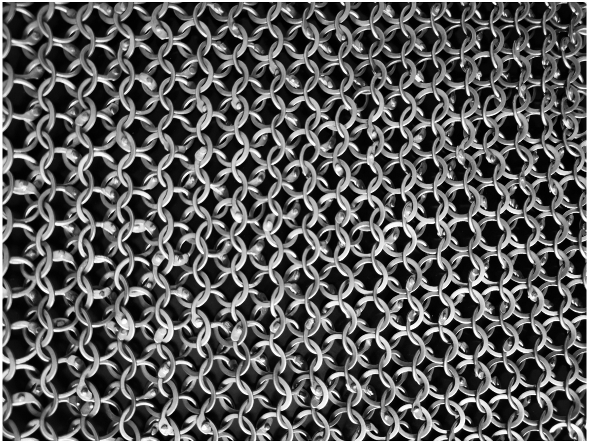
Logistic regression

We also have

$$\begin{aligned} P(- \mid w, u) = 1 - P(+ \mid w, u) &= \sigma(-\mathbf{e}_t(w) \cdot \mathbf{e}_c(u)) \\ &= \frac{1}{1 + \exp(\mathbf{e}_t(w) \cdot \mathbf{e}_c(u))} \end{aligned}$$

Well-known property of the sigmoid function.

Training



Training

For simplicity, let us consider a dataset with only one target/context pair (w, u), along with k noise words $v_1, v_2, \ldots, v_k$ (negative examples).

Negative examples are related to contrastive learning paradigm.

Skip-gram makes the simplifying assumption that all (positive and negative) context words are **independent**. Then the log-likelihood of the data is

$$
\begin{aligned}
LL_w &= \log\left(P(+ \mid w, u) \prod_{j=1}^{k} P(- \mid w, v_j) \right) \\
&= \log P(+ \mid w, u) + \sum_{j=1}^{k} \log P(- \mid w, v_j)
\end{aligned}
$$

Training

$$
\begin{aligned}
LL_w &= \log P(+ \mid w, u) + \sum_{j=1}^{k} \log(1 - P(+ \mid w, v_j)) \\
&= \log \sigma(\mathbf{e}_t(w) \cdot \mathbf{e}_c(u)) + \sum_{j=1}^{k} \log(1 - \sigma(\mathbf{e}_t(w) \cdot \mathbf{e}_c(v_j)))
\end{aligned}
$$

We can alternatively minimise the inverse of LL_w, which in case of the logistic regression is the **cross-entropy** loss function

$$
L_{\text{CE}} = -\log \sigma(\mathbf{e}_t(w) \cdot \mathbf{e}_c(u)) - \sum_{j=1}^{k} \log(1 - \sigma(\mathbf{e}_t(w) \cdot \mathbf{e}_c(v_j)))
$$

We are omitting term $\frac{1}{(k+1)}$, regarding it as a constant.

Training

The model parameters are the two $\mathbb{R}^{|V|\times d}$ matrices with the target and context embeddings $\mathbf{e}_t(w), \mathbf{e}_c(w)$, $w \in V$, which are randomly initialised.

By making an **update** to minimise L_{CE} we

- force an increase in similarity (dot product) between $\mathbf{e}_t(w)$ and $\mathbf{e}_c(u)$
- force a decrease in similarity between $\mathbf{e}_t(w)$ and $\mathbf{e}_c(v_j)$, for all of the noise words v_j.

When several target words are involved, L_{CE} is not convex and there are several local minima.

Training

We train the model with **stochastic gradient descent**, as usual for logistic regression.

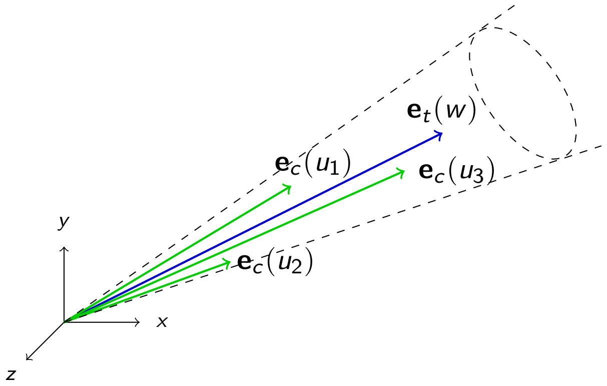
Textbook derives the gradient for the cross-entropy loss function.

In the end, we retain the target embeddings $\mathbf{e}_t(w)$ and ignore the context embeddings $\mathbf{e}_c(w)$.

In some applications, improvement in performance are reported by using $\mathbf{e}_t(w) + \mathbf{e}_c(w)$ as the final word embeddings.

Geometric interpretation

Target vector $\mathbf{e}_t(w)$ is surrounded by its context vectors $\mathbf{e}_c(u_j)$, which form a "thin" hypercone (dashed lines)



Geometric interpretation

Let w be a target word with context words u_j's. Let also w' be a target word similar to w.

Then we have that

- $\mathbf{e}_c(u_j)$ form a thin cone surrounding vector $\mathbf{e}_t(w)$
- vector $\mathbf{e}_t(w')$ must be placed within that cone, because w' and w share their context words
- $\mathbf{e}_t(w)$ and $\mathbf{e}_t(w')$ are forced to be at a **small angle**

As a result, similar words cluster together.

With a similar reasoning, we can argue that different words repel each other.

Practical issues

Practical issues

Clean up text: convert any **punctuation** into token `<PERIOD>`.

Remove all words w such that $f(w) \leqslant D$ (usually $D = 5$):

- greatly reduces issues due to noise in the data
- improves the quality of the vector representation

Indexing: convert words in V to non-negative integers and back again, this makes your code simpler.

Usually, integers are assigned in descending frequency order (most frequent word is assigned integer 0).

Practical issues

Words that show up very often, such as ‘the’, ‘of’, and ‘for’, don’t provide much context and are often regarded as noise.

The process of discarding occurrences of these words by means of some probability distribution is called **subsampling**.

We **discard** occurrence w_i with probability given by

$$P(w_i) \quad = \quad 1 - \sqrt{\frac{t}{z(w_i)}}$$

where $z(w_i)$ is the normalized frequency of w_i, and t is a threshold parameter depending on dataset size.

Example : With 10^7 word occurrences you may use $t = 10^{-5}$. Then if $z(w_i) = 0.1$ (10^6 occurrences of w_i in dataset) we get $P(w_i) = 0.99$, that is, we leave in the data 10^4 occurrences of w_i.

Practical issues

The ratio k between the positive and the negative examples is a **hyperparameter** and must be adjusted on the development set.

The noise words are chosen according to the relative frequency estimator and an α **corrector**

$$P_\alpha(v) \quad = \quad \frac{f(v)^\alpha}{\sum_{w \in V} (f(w))^\alpha}$$

Similarly to the PPMI case, $\alpha = .75$ gives better performance in most applications.

Practical issues

Choice of hyperparameter L, specifying context window size, will be discussed later.

Once hyperparameter L is fixed, context window size can be chosen **randomly at each step**, with uniform distribution in the range $[1..L]$.

This results in **linear decay** for context words, depending on distance from target word:

- words at distance 1 from target will be available with $P = 1$
- words at distance L from target will be available with $P = \frac{1}{L}$

Practical issues

Should we estimate one embedding per word form or else estimate distinct embeddings for each **word sense**?

Intuitively, if word representations are to capture the meaning of individual words, then words with multiple meanings should have multiple embeddings.

However, in certain applications this is unnecessary, because the embedding of a word form is a linear combination of the embeddings of the underlying senses; see Arora *et al.* (2018).

A few other tricks are discussed in the word2vec original papers.

Miscellanea

FastText

FastText is an embedding that deals with unknown words and sparsity in languages with rich morphology, by using **subword models**.

FastText is an extension of word2vec.

Each word in fastText is represented by itself plus a bag of character N-grams, for bounded values of N.

A character N-gram is a sequence of N characters. We will introduce N-grams in some later lecture on language models.

FastText

Example : With $N = 3$ the word 'where' would be represented by the following strings (_ is a boundary marker)

where, _wh, whe, her, ere, re_

Then a skip-gram embedding is learned for each N-gram, and the word 'where' is represented as the sum of all of the resulting embeddings.

In **morphologically rich languages** (MRLs), words display productive internal structure. These regularity are then captured by fastText.

GloVe

Global Vectors, or **GloVe** for short, is an embedding that accounts for **global corpus statistics**, which was somehow disregarded by word2vec.

GloVe combines the intuitions of count-based models like PPMI, while also capturing the linear structures used by methods like word2vec.

In the original paper for GloVe (Pennington et al., 2014), a mathematical comparison between this embedding and word2vec is provided.

Visualizing word embeddings

Word embeddings can be used to visualise the meaning of a word w.

Most commonly used methods:

- listing the words in V with highest cosine similarity with w.

  Locality-sensitive hashing (LSH) can be used, that hashes similar input items into the same buckets with high probability.

- project the d dimensions of a word embedding down into 2 or 3 dimensions.

  t-distributed stochastic neighbor embedding (t-SNE) is used, preserving metric properties.

Semantic properties

Both sparse and dense vectors are sensible to the **context window size** used to collect counts, controlled by hyperparameter L.

We generally work with 2 to 20 words in the context.

The choice of L depends on the goals of the representation:

- small values of L provide semantically similar words with the same parts of speech.

  With a window ± 2, 'Hogwarts' provides 'Sunnydale' and 'Evernight' (other fictional schools).

- larger values of L provide words that are topically related but not similar.

  With a window ± 5, 'Hogwarts' provides 'Dumbledore', 'Malfoy' and 'half-blood' (words topically related to the Harry Potter series).

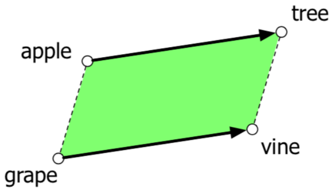
Semantic properties

Another semantic property of embeddings is their ability to capture **relational meanings**.

Word embeddings can be used to solve analogy problems, usually expressed in the form:

Apple is to tree as grape is to ________.

The **parallelogram model** can be used for solving such problems:



Fairness & bias

Unfortunately, word embeddings also reproduce the implicit biases and stereotypes that are latent in the text.

Example : On a standard dataset, embedding similarly suggests the analogy ‘father’ is to ‘doctor’ as ‘mother’ is to ‘nurse’.

This could result in what is called **allocational harm**, when a system allocates resources (jobs or credit) unfairly to different groups.

Embeddings not only reflect the statistics of their input corpora, but have also been reported to amplify bias. This is a current research topic.

Evaluation

Evaluation

Extrinsic evaluation : Use the model to be evaluated in some end-to-end application (as for instance sentiment analysis, machine translation, etc.) and measure performance.

Difficult to do reliably, time consuming.

Intrinsic evaluation : Look at performance of model in isolation, with respect to a given evaluation measure.

For word embedding, several similarity datasets; see next slide.

Evaluation

The most common evaluation metric for embedding models is extrinsic evaluation on end-to-end tasks: machine translation, sentiment analysis, etc.

Three types of intrinsic evaluation

- word **similarity**: compute a numerical score for the semantic similarity between two words;
  Example : car, automobile
- word **relatedness**: compute the degree of how much one word has to do with another word;
  Example : car, highway
- word **analogy**: comparison of two things to show their similarities;
  Example : tree : leaf :: flower : petal

Evaluation

Several datasets available for intrinsic evaluation:

- **WordSim-353** and **SimLex-999** are collection for measuring word relatedness and similarity for word pairs in isolation.
- **Stanford Contextual Word Similarity** (SCWS) and **Word-in-Context** (WiC) datasets provide pairs of words in their sentential context, offering richer evaluation scenarios.
- Several sets from **SemEval-2012 Task 2** provide analogy tasks covering morphology, lexicographic and encyclopedia relations.

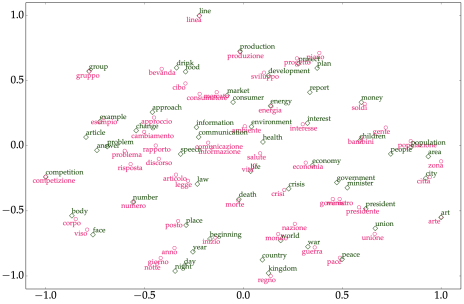
Cross-lingual word embedding

©cristinn – Adobe Stock

Cross-lingual word embedding

Cross-lingual word embeddings encode words from two or more languages in a shared high-dimensional space.

Vectors representing words with similar meaning are closely located, regardless of language.



Ruder *et al.*, 2019

Cross-lingual word embedding

Cross-lingual word embeddings

- are very useful in **comparing** the meaning of words across languages
- are key to applications such as machine translation, multilingual lexicon induction, and cross-lingual information retrieval
- enable model **transfer** between resource-rich and low-resource languages, by providing a common representation space

Cross-lingual word embedding

The class of **objective functions** minimized by most cross-lingual word embedding methods can be formulated as

$$J = \mathcal{L}^1 + \cdots + \mathcal{L}^\ell + \Omega$$

where $\mathcal{L}^i$ is the monolingual loss of the i-th language and Ω a regularization term.

Joint optimization of multiple non-convex losses is **difficult**

- most approaches optimize one loss at a time, while keeping certain variables fixed
- this step-wise approach is approximate and does not guarantee to reach even a local optimum

Cross-lingual word embedding

The choice of multilingual **parallel data sources** is more important for the model performance than the actual underlying architecture.

Methods differ along the following two dimensions.

- Type of text **alignment**: at the level of words, sentences, or documents, which introduce stronger or weaker supervision.
- Type of text **comparability**: exact translations, weak translation, or similar meaning.

Cross-lingual word embedding

Extending the learning process from bilingual to multilingual settings **improves** results, as reported on many standard tasks.

Variation in ambiguity: ambiguity for word *w* in one language does not transfer to the translation of *w* in another language.

The general hypothesis is that variation in ambiguity acts as a form of naturally occurring **supervision**.